THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Họ tên: Khương Thanh Bình

MSSV:3123411034

Bài Tập Ngăn Xếp Và Hàng Đợi

**BÀI 001**

• Cho một cây nhị phân có nút gốc là Root, mỗi nút trong cây chứa một số

nguyên:

a) Viết chương trình tính trung bình cộng các nút trong cây.

b) Viết chương trình tính trung bình cộng các số dương trong cây.

c) Viết chương trình tính trung bình cộng các số âm trong cây.

d) Viết chương trình tính tính tỉ số R=a/b. Với a là tổng các nút có giá trị

dương, b là tổng các nút có giá trị âm**.**

CODE

#include<iostream>

using namespace std;

struct node {

int info;

int count;

struct node\* pLeft;

struct node\* pRight;

};

typedef struct node NODE;

typedef NODE\* TREE;

TREE root = NULL;

void Init(TREE& Root) {

Root = NULL;

}

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->count = 1;

p->pLeft = NULL;

p->pRight = NULL;

return p;

}

void Insert(TREE& p, int theKey) {

if (p == NULL) {

p = CreateNode(theKey);

}

else if (p->info > theKey) {

Insert(p->pLeft, theKey);

}

else if (p->info == theKey) {

p->count = p->count + 1;

}

else {

Insert(p->pRight, theKey);

}

}

int DemNode(TREE Root) {

if (Root == NULL) return 0;

int a = DemNode(Root->pLeft);

int b = DemNode(Root->pRight);

return (a + b + 1);

}

int TongNode(TREE Root) {

if (Root == NULL) return 0;

int a = TongNode(Root->pLeft);

int b = TongNode(Root->pRight);

return (a + b + Root->info);

}

float TrungBinhCong(TREE Root) {

int s = TongNode(Root);

int dem = DemNode(Root);

if (dem == 0) return 0;

return (float)s / dem;

}

int DemDuong(TREE Root) {

if (Root == NULL) return 0;

int a = DemDuong(Root->pLeft);

int b = DemDuong(Root->pRight);

if (Root->info > 0)

return (a + b + 1);

return (a + b);

}

int TongDuong(TREE Root) {

if (Root == NULL) return 0;

int a = TongDuong(Root->pLeft);

int b = TongDuong(Root->pRight);

if (Root->info > 0)

return (a + b + Root->info);

return (a + b);

}

float TrungBinhDuong(TREE Root) {

int s = TongDuong(Root);

int dem = DemDuong(Root);

if (dem == 0) return 0;

return (float)s / dem;

}

int TongAm(TREE Root) {

if (Root == NULL) return 0;

int a = TongAm(Root->pLeft);

int b = TongAm(Root->pRight);

if (Root->info < 0)

return (a + b + Root->info);

return (a + b);

}

int DemAm(TREE Root) {

if (Root == NULL) return 0;

int a = DemAm(Root->pLeft);

int b = DemAm(Root->pRight);

if (Root->info < 0)

return (a + b + 1);

return (a + b);

}

float TrungBinhCongAm(TREE Root) {

int s = TongAm(Root);

int dem = DemAm(Root);

if (dem == 0) return 0;

return (float)s / dem;

}

float TinhTySo(TREE Root) {

int a = TongDuong(Root);

int b = TongAm(Root);

if (b == 0) return 0;

return (float)a / b;

}

int main() {

TREE Root;

Init(Root);

int arr[] = { 10, -5, 15, -10, 0, 20, -3, 7, 10 };

int n = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < n; i++) {

Insert(Root, arr[i]);

}

cout << "Trung binh cong cac nut: " << TrungBinhCong(Root) << endl;

cout << "Trung binh cong cac nut duong: " << TrungBinhDuong(Root) << endl;

cout << "Trung binh cong cac nut am: " << TrungBinhCongAm(Root) << endl;

cout << "Ti so R = a / b: " << TinhTySo(Root) << endl;

return 0;

}

**-** Chạy code

->A black background with white text

AI-generated content may be incorrect.

**BÀI 002**

• Cho cây như hình 1, cho trước nút p

A diagram of a network

AI-generated content may be incorrect.

• Hãy viết các câu lệnh cần thiết để chuyển cây sang dạng biểu diễn của hình 2

CODE

#include<iostream>

using namespace std;

struct node {

int info;

struct node\* pLeft;

struct node\* pRight;

};

typedef struct node NODE;

typedef NODE\* TREE;

TREE root = NULL;

void Init(TREE& Root) {

Root = NULL;

}

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = NULL;

p->pRight = NULL;

return p;

}

void Insert(TREE& p, int theKey) {

if (p == NULL) {

p = CreateNode(theKey);

}

else if (p->info > theKey) {

Insert(p->pLeft, theKey);

}

else if (p->info < theKey) {

Insert(p->pRight, theKey);

}

}

void InOrder(TREE Root) {

if (Root != NULL) {

InOrder(Root->pLeft);

cout << Root->info << " ";

InOrder(Root->pRight);

}

}

TREE RotateRight(TREE p) {

if (p == NULL || p->pLeft == NULL) return p;

NODE\* temp = p->pLeft;

p->pLeft = p->pRight;

p->pRight = temp;

temp = p->pLeft->pLeft;

p->pLeft->pLeft = p->pLeft->pRight;

p->pLeft->pRight = temp;

return p;

}

int main() {

TREE Root;

Init(Root);

int arr[] = { 10, 5, 15, 3, 7, 9, 12, 18, 20 };

int n = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < n; i++) {

Insert(Root, arr[i]);

}

cout << "Cay truoc khi xoay: ";

InOrder(Root);

cout << endl;

Root = RotateRight(Root);

cout << "Cay sau khi xoay: ";

InOrder(Root);

cout << endl;

return 0;

}

- Chạy code

-> 

**BÀI 003**

Cho cây nhị phân tìm kiếm như hình vẽ. Hãy cho biết thứ tự các nút thêm

vào cây sao cho để có được cấu trúc này? (Giả sử lúc đầu cây rỗng).

A diagram of a network

AI-generated content may be incorrect.

Nếu kết quả của phép duyệt cây trên là: 3, 7, 9, 5, 12, 20, 18, 15, 10. Hãy

cho biết người ta đã áp dụng phép duyệt nào?

--GIẢI--

Thứ tự các nút thêm vào cây để có được cây cấu trúc như trên là:

• Cách 01: 10, 5, 15, 3, 9, 12, 18, 7, 20.

• Cách 02: 10, 5, 3, 9, 7, 15, 12, 18, 20.

• Cách 03: 10, 15, 18, 20, 12, 5, 9, 7, 3.

Nếu kết quả của phép duyệt cây trên là: 3, 7, 9, 5, 12, 20, 18, 15, 10.

Hãy cho biết người ta đã áp dụng phép duyệt nào?

• Đó là phép duyệt cây theo phương pháp: LRN.

**BÀI 004**

Cho một cây nhị phân tìm kiếm với nút gốc là Root, giá trị lưu trữ

tại mỗi nút là một số nguyên (int). Hãy viết hàm tìm phần tử nhỏ

nhất và lớn nhất trong cây.

A diagram of a network

AI-generated content may be incorrect.

CODE

#include<iostream>

using namespace std;

struct node {

int info;

struct node\* pLeft;

struct node\* pRight;

};

typedef struct node NODE;

typedef NODE\* TREE;

TREE root = NULL;

void Init(TREE& Root) {

Root = NULL;

}

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = NULL;

p->pRight = NULL;

return p;

}

void Insert(TREE& p, int theKey) {

if (p == NULL) {

p = CreateNode(theKey);

}

else if (p->info > theKey) {

Insert(p->pLeft, theKey);

}

else if (p->info < theKey) {

Insert(p->pRight, theKey);

}

}

void InOrder(TREE Root) {

if (Root != NULL) {

InOrder(Root->pLeft);

cout << Root->info << " ";

InOrder(Root->pRight);

}

}

NODE\* NhoNhat(TREE Root) {

if (Root == NULL) return NULL;

NODE\* lc = Root;

while (lc->pLeft)

lc = lc->pLeft;

return lc;

}

NODE\* LonNhat(TREE Root) {

if (Root == NULL) return NULL;

NODE\* lc = Root;

while (lc->pRight)

lc = lc->pRight;

return lc;

}

int main() {

TREE Root;

Init(Root);

int arr[] = { 10, 5, 15, 3, 7, 9, 12, 18, 20 };

int n = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < n; i++) {

Insert(Root, arr[i]);

}

cout << "Cay: ";

InOrder(Root);

cout << endl;

NODE\* minNode = NhoNhat(Root);

NODE\* maxNode = LonNhat(Root);

if (minNode) cout << "Phan tu nho nhat: " << minNode->info << endl;

if (maxNode) cout << "Phan tu lon nhat: " << maxNode->info << endl;

return 0;

}

- Chạy code

->A black background with white text

AI-generated content may be incorrect.

**BÀI 005**

Cho một cây nhị phân có cấu trúc nút là NODE hãy:

• Viết hàm để tính tổng số nút có một nhánh con (con trái HAY con phải) bằng cách dùng thuật toán duyệt nút gốc giữa NLR;

• Thiết lập một công thức đệ quy để thực hiện yêu cầu của câu trên. Cài đặt công thức này thành hàm.

• Câu b: Số lượng nút một con trong cây nhị phân bằng 1 cộng số lượng nút một con trong cây nhị phân con trái cộng số lượng nút một con trong cây nhị phân con phải nếu nút gốc có một con. Ngược lại số lượng nút một con trong cây nhị phân bằng số lượng nút một con trong cây nhị phân con trái cộng số lượng nút một con trong cây nhị phân con phải.

CODE

#include<iostream>

using namespace std;

struct node {

int info;

struct node\* pLeft;

struct node\* pRight;

};

typedef struct node NODE;

typedef NODE\* TREE;

TREE root = NULL;

void Init(TREE& Root) {

Root = NULL;

}

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = NULL;

p->pRight = NULL;

return p;

}

void Insert(TREE& p, int theKey) {

if (p == NULL) {

p = CreateNode(theKey);

}

else if (p->info > theKey) {

Insert(p->pLeft, theKey);

}

else if (p->info < theKey) {

Insert(p->pRight, theKey);

}

}

void InOrder(TREE Root) {

if (Root != NULL) {

InOrder(Root->pLeft);

cout << Root->info << " ";

InOrder(Root->pRight);

}

}

int DemMotCon(TREE t) {

if (t == NULL)

return 0;

if ((t->pLeft && !t->pRight) || (!t->pLeft && t->pRight))

return 1 + DemMotCon(t->pLeft) + DemMotCon(t->pRight);

return DemMotCon(t->pLeft) + DemMotCon(t->pRight);

}

int main() {

TREE Root;

Init(Root);

int arr[] = { 10, 5, 15, 3, 7, 9, 12, 18, 20 };

int n = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < n; i++) {

Insert(Root, arr[i]);

}

cout << "Cay: ";

InOrder(Root);

cout << endl;

cout << "So luong nut co mot con: " << DemMotCon(Root) << endl;

return 0;

}

- Chạy code

-> 

**BÀI 007**

Hãy mô tả ngắn gọn sự giống và khác nhau giữa hai cấu trúc Cây Nhị Phân tìm kiếm và Danh Sách Liên Kết Đơn.

--GIẢI--

• Giống nhau:

• CTDL động.

• Các thao tác cơ bản Thêm, Xóa, Cập Nhật được thực hiện một cách linh hoạt.

• Khác nhau:

• Dữ liệu trên cây NPTK được tổ chức và dslk đơn thì không.

• Chi phí tìm kiếm, thêm trên cây nhanh hơn trên dslk đơn.

**BÀI 008**

• Định nghĩa hàm duyệt và xuất cây nhị phân các

số thực ra tập tin nhị phân data.out theo phương

pháp LNR.

CODE

#include<iostream>

#include<fstream>

using namespace std;

struct node {

float info;

struct node\* pLeft;

struct node\* pRight;

};

typedef struct node NODE;

typedef NODE\* TREE;

TREE root = NULL;

void Init(TREE& Root) {

Root = NULL;

}

NODE\* CreateNode(float x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = NULL;

p->pRight = NULL;

return p;

}

void Insert(TREE& p, float theKey) {

if (p == NULL) {

p = CreateNode(theKey);

}

else if (p->info > theKey) {

Insert(p->pLeft, theKey);

}

else if (p->info < theKey) {

Insert(p->pRight, theKey);

}

}

void InOrder(TREE Root) {

if (Root != NULL) {

InOrder(Root->pLeft);

cout << Root->info << " ";

InOrder(Root->pRight);

}

}

void LNR(TREE t, FILE\* fp) {

if (t == NULL)

return;

LNR(t->pLeft, fp);

fwrite(&t->info, sizeof(float), 1, fp);

LNR(t->pRight, fp);

}

int Xuat(const char\* filename, TREE t) {

FILE\* fp = fopen(filename, "wb");

if (fp == NULL)

return 0;

LNR(t, fp);

fclose(fp);

return 1;

}

void DocFile(const char\* filename) {

FILE\* fp = fopen(filename, "rb");

if (fp == NULL) {

cout << "Khong the mo file!" << endl;

return;

}

float value;

cout << "Cac gia tri trong file: ";

while (fread(&value, sizeof(float), 1, fp)) {

cout << value << " ";

}

cout << endl;

fclose(fp);

}

int main() {

TREE Root;

Init(Root);

float arr[] = { 10.5, 5.2, 15.8, 3.1, 7.4, 9.6, 12.3, 18.9, 20.0 };

int n = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < n; i++) {

Insert(Root, arr[i]);

}

cout << "Cay: ";

InOrder(Root);

cout << endl;

if (Xuat("data.out", Root)) {

cout << "Da xuat cay vao file data.out thanh cong!" << endl;

}

else {

cout << "Xuat file that bai!" << endl;

}

DocFile("data.out");

return 0;

}

- Chạy code

-> A black background with white text

AI-generated content may be incorrect.

**BÀI 009**

• Định nghĩa hàm duyệt và xuất cây nhị phân các số thực ra tập tin nhị phân data.out theo phương pháp NLR.

CODE

#include<iostream>

#include<fstream>

using namespace std;

struct node {

float info;

struct node\* pLeft;

struct node\* pRight;

};

typedef struct node NODE;

typedef NODE\* TREE;

TREE root = NULL;

void Init(TREE& Root) {

Root = NULL;

}

NODE\* CreateNode(float x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = NULL;

p->pRight = NULL;

return p;

}

void Insert(TREE& p, float theKey) {

if (p == NULL) {

p = CreateNode(theKey);

}

else if (p->info > theKey) {

Insert(p->pLeft, theKey);

}

else if (p->info < theKey) {

Insert(p->pRight, theKey);

}

}

void InOrder(TREE Root) {

if (Root != NULL) {

InOrder(Root->pLeft);

cout << Root->info << " ";

InOrder(Root->pRight);

}

}

void NLR(TREE t, FILE\* fp) {

if (t == NULL)

return;

fwrite(&t->info, sizeof(float), 1, fp);

NLR(t->pLeft, fp);

NLR(t->pRight, fp);

}

int Xuat(const char\* filename, TREE t) {

FILE\* fp = fopen(filename, "wb");

if (fp == NULL)

return 0;

NLR(t, fp);

fclose(fp);

return 1;

}

void DocFile(const char\* filename) {

FILE\* fp = fopen(filename, "rb");

if (fp == NULL) {

cout << "Khong the mo file!" << endl;

return;

}

float value;

cout << "Cac gia tri trong file: ";

while (fread(&value, sizeof(float), 1, fp)) {

cout << value << " ";

}

cout << endl;

fclose(fp);

}

int main() {

TREE Root;

Init(Root);

float arr[] = { 10.5, 5.2, 15.8, 3.1, 7.4, 9.6, 12.3, 18.9, 20.0 };

int n = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < n; i++) {

Insert(Root, arr[i]);

}

cout << "Cay duyet truoc (NLR): ";

InOrder(Root);

cout << endl;

if (Xuat("data.out", Root)) {

cout << "Da xuat cay vao file data.out thanh cong!" << endl;

}

else {

cout << "Xuat file that bai!" << endl;

}

DocFile("data.out");

return 0;

}

- Chạy code

-> A black background with white text

AI-generated content may be incorrect.